TUGAS AKHIR – TE 141599

DESAIN DAN IMPLEMENTASI PROTOKOL MODBUS UNTUK SISTEM ANTRIAN TERINTEGRASI PADA PELAYANAN SURAT IZIN MENGEMUDI (SIM) DI KEPOLISIAN RESORT

Septian Dwi Chandra
NRP 2214105025

Dosen Pembimbing
Ir. Hendra Kusuma M.Eng.
Suwito, ST., MT

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE 141599

DESIGN AND IMPLEMENTATION OF MODBUS PROTOCOL FOR INTEGRATED QUEUE SYSTEM IN DRIVING LICENCE SERVICE OF RESORT POLICE OFFICE

Septian Dwi Chandra
NRP 2214105025

Supervisor
Ir. Hendra Kusuma , M.Eng., Sc.
Suwito, ST., MT

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

DESAIN DAN IMPLEMENTASI PROTOKOL MODBUS UNTUK SISTEM ANTRIAN TERINTEGRASI PADA PELAYANAN SURAT IZIN MENGEMUDI (SIM) DI KEPOLISIAN RESORT

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I,	Dosen Pembimbing II,
Ir. Hendra Kusuma, M.Eng., Sc.	Suwito, ST., MT.
NIP: 196409021989031003	NIP: 198101052005011004

SURABAYA
JUNI, 2016

--Halaman ini sengaja dikosongkan—

DESAIN DAN IMPLEMENTASI PROTOKOL MODBUS UNTUK SISTEM ANTRIAN TERINTEGRASI PADA PELAYANAN SURAT IZIN MENGEMUDI (SIM) DI KEPOLISIAN RESORT

Nama : Septian Dwi Chandra
Pembimbing : Ir. Hendra Kusuma, M.Eng., Sc.
Suwito, ST, MT

ABSTRAK

Pelayanan pembuatan SIM terdiri dari pendaftaran SIM baru atau perpanjangan, ujian tes tulis, ujian tes lapangan, pas foto, dan pembayaran. Di semua proses tersebut, pendaftar dipanggil satu per satu menggunakan pengeras suara sesuai dengan urutan dokumen yang ditumpuk oleh pemohon.

Untuk meningkatkan kualitas layanan, dirancanglah sebuah sistem antrian terintegrasi yang berfungsi untuk melakukan pemanggilan yang lebih efisien. Pada perancangan ini mikrokontroler dimanfaatkan sebagai perangkat slave untuk berkomunikasi dengan PC sebagai master. *Slave* akan mengirimkan data berupa nomor urut antrian yang kemudian akan ditampilkan di layar LCD sehingga proses antrian dapat diawasi baik oleh pemohon maupun petugas. Pemohon hanya perlu mengambil nomor urut, kemudian menunggu panggilan yang ditampilkan di layar LCD utama. Petugas hanya perlu menekan tombol tanpa perlu mengecek urutan berkas dan memanggil nama satu-per satu. Dengan begitu pelayanan akan jauh lebih cepat dan mudah.

Untuk menjalankan sistem tersebut diperlukan perangkat yang saling terkoneksi di tiap ruang pelayanan. Dalam hal ini dipilihlah Modbus sebagai protokolnya. Modbus merupakan protokol komunikasi yang bebas royalti dan bebas untuk dikembangkan. Dari beberapa jenis Modbus yang ada, Modbus RTU memiliki kriteria yang dibutuhkan. Selain kebutuhan hardware yang lebih sederhana, pengaplikasian protokol komunikasinya juga sangat memungkinkan untuk diterapkan pada sebuah mikrokontroler sehingga diharapkan dapat menciptakan sebuah sistem antrian dengan biaya rendah namun memiliki kinerja yang sesuai dengan kebutuhan sistem.

Kata Kunci – Modbus RTU, Mikrokontroler AVR, sistem antrian, interfacing.

--Halaman ini sengaja dikosongkan—

DESIGN AND IMPLEMENTATION OF MODBUS PROTOCOL FOR INTEGRATED QUEUE SYSTEM IN DRIVING LICENCE SERVICE OF RESORT POLICE OFFICE

Name **:** Septian Dwi Chandra
Supervisor **:** Ir. Hendra Kusuma, M.Eng., Sc.
Suwito, ST, MT

ABSTRACT

Services of renewal Driving Licence are consists of a writing exam, exam field exam, photo room, and payments. In all these processes, applicants are called one by one using a loudspeaker in accordance with the order of the document are stacked by the applicant.

To improve the quality of the service, the idea is to create an integrated queuing system for more effective and efficience service. This design is using microcontroller which used as a slave device to communicate with a PC as a master. Slave sends data of the serial number of a queue which will be displayed on the LCD screen so that the queue process can be monitored either by the applicant or officer. Applicant just needs to take the serial number, then wait for the call that displayed on the main LCD screen. Officers only need to press a button without the need to check the stacked document and calling applicant's name one by one. With so services will be much faster and easier.

This system required a device that connected to each other in each service room. In this case Modbus protocol was chosen. Modbus is a communication protocol that royalty-free and easy to deploy. Between several types of existing Modbus, Modbus RTU has the meet the criteria. In addition to simplify the hardware requirements, the application of the communication protocol is also very possible to be applied to a microcontroller. This design is expected to create a queuing system with low-cost but have a good performance according to the system needs.

Keywords *– Modbus RTU, Microcontroller AVR, Queue System, Interfacing.*

.

--Halaman ini sengaja dikosongkan--

KATA PENGANTAR

Segala puji syukur penulis panjatkan atas kehadirat Allah SWT yang selalu memberikan rahmat serta hidayah-Nya sehingga Tugas Akhir ini dapat terselesaikan dengan baik. Shalawat serta salam selalu dilimpahkan kepada Rasulullah Muhammad SAW, keluarga, sahabat, dan umat muslim yang senantiasa meneladani beliau.

Pada kesempatan ini penulis ingin mengucapkan ucapan terimakasih yang sebesar- besarnya kepada beberapa pihak yang telah memberikan dukungan selama proses pengerjaan tugas akhir ini, antara lain:

1. Keluarga penulis Subiyanto, Ibu Rokhani, dan seluruh keluarga yang selalu memberikan doa, motivasi, semangat, perhatian dan kasih sayangnya.
2. Bapak Ir. Hendra Kusuma, M.Eng.,Sc. dan Bapak Suwito, ST, MT, selaku dosen pembimbing yang selalu memberikan arahan dan ilmu dalam penyelesaian Tugas Akhir ini.
3. Seluruh dosen bidang studi Elektronika Jurusan Teknik Elektro FTI ITS.
4. Inas Ratnaning Zafirah, Rahmad Hidayat, Erwan Aprillian, Ervan Wahyudi, Fadli Husaini, Nasyith Hanarur, Kriwul dan teman-teman D3 maupun S1 Teknik Elektro, serta banyak pihak yang tidak dapat disebutkan satu-persatu atas kebersamaannya.

Penulis menyadari bahwa dalam Tugas Akhir ini terdapat banyak kekurangan. Akhir kata semoga melalui tulisan ini dapat bermanfaat dan dapat berbagi ilmu bagi pembacanya. Amin.

Surabaya, Mei 2016

Penulis

--Halaman ini sengaja dikosongkan--

DAFTAR ISI

DAFTAR GAMBAR

--Halaman ini sengaja dikosongkan--

DAFTAR TABEL

--Halaman ini sengaja dikosongkan--

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pelayanan publik merupakan salah satu tugas penting yang tidak dapat diabaikan oleh Kepolisian oleh sebab itu perlu ada perencanaan yang baik sehingga pelayanan pada masyarakat menjadi lebih baik dan efisien. Salah satunya adalah pelayanan pembuatan SIM (Surat Izin Mengemudi), yang merupakan salah satu pelayanan dasar administratif dalam penggunaan kendaraan bermotor.

Pelayanan pembuatan SIM terdiri dari pendaftaran SIM baru atau perpanjangan, ujian tes tulis, ujian tes lapangan, pas foto, dan pembayaran. Di semua proses tersebut, pendaftar dipanggil satu per satu menggunakan pengeras suara.

Dari proses tersebut, efisiensi dan kualitas pelayanan dapat ditingkatkan. Salah satunya dengan sistem antrian otomatis, dimana petugas hanya perlu menekan tombol dan nomor urut akan disesuaikan secara otomatis melalui software sehingga akan lebih menghemat tenaga dan meminimalisir kesalahan pemanggilan urutan yang biasanya terjadi. Selain itu pendaftar SIM juga akan lebih nyaman karena dapat melihat langsung urutan nomor antrian yang sedang berlangsung.

Dalam tugas akhir ini didesain sebuah sistem antrian pada pelayanan SIM di Kepolisian Resort (POLRES) Pasuruan yang memanfaatkan protokol Modbus RTU sebagai media komunikasi antar ruangan untuk memroses urutan antrian. Di tiap ruangan akan disediakan kotak tombol (slave) untuk memanggil nomor antrian selanjutnya yang terhubung dengan kontrol utama (master) berupa PC yang menampilkan display antrian.

Sistem ini terintegrasi dengan PC *back office* sebagai *slave* yang terhubung dengan sistem antrian utama. PC back office ini digunakan untuk mengawasi dan mengkoleksi data antrian di jam kerja. Fungsinya adalah untuk mempermudah petugas dalam melakukan analisa dan evaluasi pelayanan. Sehingga kualitas pelayanan dapat lebih ditingkatkan berdasarkan data yang didapat dari statistik antrian.

Dengan adanya sistem ini tidak menutup kemungkinan dapat diintegrasikan dengan pelayanan lainnya yang ada di Kepolisian Resort, seperti pengurusan BPKB, STNK, dan sebagainya.

1.2 Rumusan Masalah

Sehubungan dengan latar belakang yang telah dijelaskan sebelumnya, terdapat beberapa masalah yang akan dibahas antara lain sebagai berikut :

1. Efisiensi sistem antrian.
2. Peningkatan kualitas pelayanan SIM di Kepolisian Resort.
3. Keandalan protokol Modbus RTU pada aplikasi non-industri.

1.3 Batasan Masalah

Batasan masalah yang akan dibahas dalam Tugas Akhir ini adalah :

1. Mengirim, menerima, dan mengolah data berbasis komunikasi Modbus.
2. Perancangan sistem antrian di unit Pelayanan SIM POLRES.
3. Implementasi Modbus menggunakan mikrokontroler AVR.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah untuk merancang sebuah sistem antrian di unit Pelayanan Pembuatan Surat Izin Mengemudi (SIM) untuk menhasilkan pelayanan masyarakat yang lebih baik dan efisien.

1.5 Metodologi

Metodologi yang digunakan dalam penyusunan tugas akhir ini sebagai berikut :

1. Studi literatur dan diskusi, yaitu studi yang bersumber pada jurnal-jurnal, buku referensi dan *datasheet* komponen yang digunakan dalam Tugas Akhir ini serta berdiskusi dengan dosen pembimbing dan narasumber.
2. Perancangan dan realisasi alat, pada tahap ini dimulai dari perancangan kontroler untuk perangkat slave hingga pemrograman mikrokontroler maupun interface hingga fungsi-fungsi sistem yang diinginkan tercapai.
3. Tahap pengujian sistem dan analisa, pada tahapan ini dimulai debugging dengan terlebih dahulu menguji respon tiap komponen. Kemudian menggabungkannya menjadi sistem utuh hingga respon sistem telah sesuai.
4. Pembuatan laporan mengacu pada perancangan dan realisasi alat, serta hasil dari pengujian alat.

1.6 Sistematika Penulisan

Sistematika penulisan pada tugas akhir ini dibagi menjadi beberapa bab dengan rincian :

BAB I : PENDAHULUAN

Berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan.

BAB II : TEORI PENUNJANG

Berisi tentang sistem komunikasi Modbus Serial, MTU (Master Terminal Unit) yang digunakan pada sistem dan RTU (Remote Terminal Unit) yg berfungsi sebagai slave yang digunakan dalam sistem.

BAB III : PERANCANGAN DAN REALISASI ALAT

Berisi tentang tahap-tahap perancangan perangkat hardware dan software interface.

BAB IV : PENGUJIAN DAN ANALISA SISTEM

Bab ini membahas mengenai pengujian dari rancangan yang telah diimplementasikan pada sistem antrian berbasis Modbus.

BAB V : PENUTUP

Berisi tentang kesimpulan dan saran yang diperoleh dalam Tugas Akhir ini.

1.7 Relevansi

Adapun manfaat yang diharapkan dengan tugas akhir ini adalah sistem antrian terintegrasi yang dirancang dapat benar-benar diimplementasikan dengan baik serta mampu meningkatkan kualitas pelayanan pembuatan Surat Izin Mengemudi (SIM) di Kepolisian Resort terutama Resort Pasuruan.

Hasil yang dicapai diharapkan dapat menjadi salah satu referensi dalam pengembangan sistem antrian dengan pengolahan informasi yang lebih jauh.

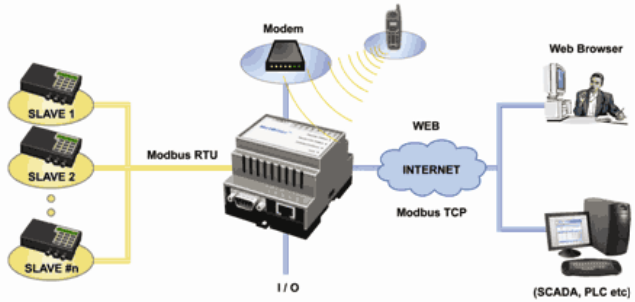
--Halaman ini sengaja dikosongkan--

BAB II
TEORI PENUNJANG

2.1 Protokol Modbus

Modbus adalah protokol komunikasi serial yang diterbitkan oleh Modicon pada 1979 untuk diaplikasikan pada programmable logic controller (PLC). Kemudian protokol ini telah menjadi standar de facto protokol komunikasi di industri, dan sekarang Modbus merupakan protokol komunikasi dua-arah yang paling umum digunakan sebagai media penghubung dengan perangkat industri atau media elektronik lainnya dengan komputer.

Modbus adalah sebuah protokol terbuka, yang berarti bahwa itu gratis bagi produsen untuk membangun ke dalam peralatan mereka tanpa harus membayar royalti. Hal ini telah menjadi protokol komunikasi standar dalam industri, dan sekarang cara yang paling umum tersedia untuk menghubungkan perangkat elektronik industri. Modbus biasanya digunakan untuk mengirimkan sinyal dari perangkat instrumentasi dan kontrol kembali ke controller utama atau sistem pengumpulan data, misalnya sistem yang mengukur suhu dan kelembaban dan mengkomunikasikan hasilnya ke komputer.



Gambar 2.1 Ilustrasi jaringan protokol modbus

Modbus sering digunakan untuk menghubungkan komputer pengawasan dengan unit terminal remote (RTU) kontrol pengawasan dan akuisisi data (SCADA) sistem. Versi protokol Modbus ada untuk baris serial (Modbus RTU dan Modbus ASCII) dan untuk Ethernet (Modbus TCP). Alasan utama penggunaan Modbus secara ekstensif sebagai protokol komunikasi adalah :

a) Modbus diterbitkan sebagai *open protocol* dan bebas royalti
b) Modbus relatif mudah untuk digabungkan dengan jaringan industri
c) Modbus melakukan transfer data *raw bits* atau *words* tanpa membatasi jenis vendor atau jenis merk pabrikan perangkat industri yang digunakan.

Modbus memungkinkan adanya komunikasi dua-jalur antar perangkat yang terhubung ke jaringan yang sama, misalnya suatu sistem yang mengukur suhu dan kelembaban dan mengkomunikasikan hasilnya ke komputer (HMI). Modbus sering digunakan untuk menghubungkan s*opervisory computer* dengan *remote terminal unit* (RTU) *supervisory control* dan sistim akuisisi data (SCADA).

2.1.1 Komunikasi dan perangkat Modbus

Setiap perangkat yang diinginkan untuk berkomunikasi via protokol Modbus harus diberi alamat yang unik atau tidak boleh sama dengan alamat perangkat lainnya. Dalam komunikasi serial dan jaringan MB+ hanya node yang ditugaskan sebagai Master saja yang dapat memulai perintah, berbeda halnya dengan Ethernet, perangkat manapun dapat mengirimkan perintah Modbus, walaupun biasanya hanya satu perangkat master yang melakukannya.

Perintah Modbus berisi alamat Modbus perangkat yang ingin dituju atau yang ingin diminta berkomunikasi.. Hanya perangkat yang dimaksudkan akan bertindak atas perintah, meskipun perangkat lain mungkin juga menerima pesan/perintah tersebut (pengecualian adalah perintah *broadcastable* khusus dikirim ke *node* 0 yang bertindak tapi tidak diakui). Semua perintah pada Modbus mengandung pemeriksaan informasi, untuk memastikan bahwa perintah yang datang tidak rusak atau error. The Perintah dasar pada Modbus dapat memerintahkan sebuah RTU untuk mengubah nilai salah satu kontrol, register atau membaca sebuah port Input/Output, serta sekaligus memerintahkan perangkat untuk mengirimkan kembali satu atau lebih nilai yang terkandung dalam register yang diakses atau dirubah tersebut.

Ada banyak modem dan *gateway* yang didukung oleh Modbus, karena memang Modbus merupakan protokol yang sangat sederhana dan sering disalin oleh pabrikan-pabrikan perangkat elektronik dan jaringan. Beberapa dari mereka, ada yang secara khusus membuat perangkat yang dirancang untuk protokol ini. Implementasi yang berbeda menggunakan

kabel, komunikasi nirkabel dan bahkan SMS atau GPRS. Masalah klasik para desainer sistem monitoring dengan jaringan nirkabel/wireless, SMS dan GPRS adalah bahwa sistem yang mereka buat harus mampu mencakup latensi tinggi dan mengatasi masalah waktu.

2.1.2 Penyimpanan data pada Modbus

Pada protokol Modbus terdapat 4 buah jenis akses data dengan panjang masing-masing 16 bit.

Tabel 2.1 Register pada Modbus RTU

Nama Register	Tipe Objek	Akses	Keterangan
Coils	Single Bit	Read-Write	Master bisa membaca maupun menulis data coil
Discrete Input	Single Bit	Read only	Data hanya bisa dirubah oleh slave
Input Register	16-bit Word	Read only	Data hanya bisa dirubah oleh slave
Holding Register	16-bit Word	Read-Write	Master bisa membaca maupun menulis register

a) *Coil*

Register ini merupakan register untuk menyimpan nilai diskrit on atau off. Panjang data setiap register adalah 16 bit. Dimana untuk data yang digunakan untuk mengaktifkan *coil* (ON) bernilai 0xFF00. Sedangkan untuk menonaktifkan *coil* (OFF) bernilai 0x0000. Data ini dapat disimpan di register 00000 sampai 09999.

b) *Input relay*

Kebalikan dengan *coil*, *input relay* digunakan untuk mengetahui status relay apakah sedang dalam kondisi ON atau OFF. *Input relay* bersifat read only bagi master dan hanya bisa dirubah oleh slave yang bersangkutan. Data tersebut disimpan di register 10001 sampai 19999.

c) *Input* Register

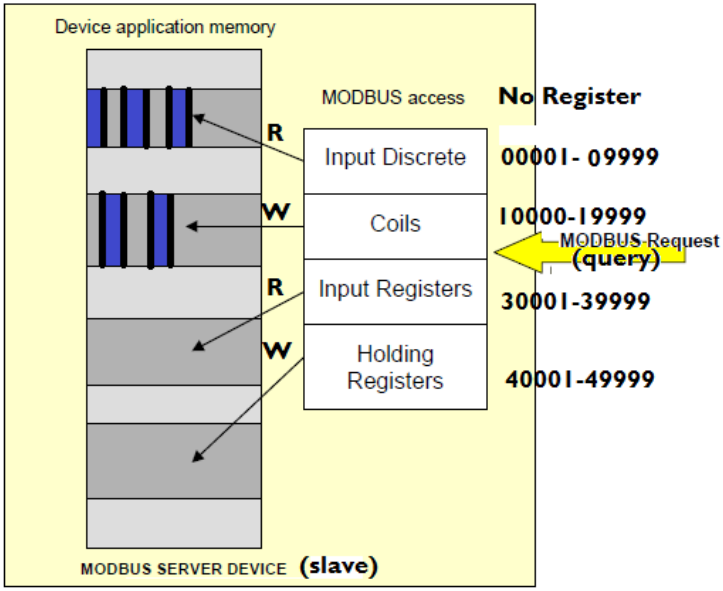
Input regsiter digunakan untuk menyimpan data analog dgn range nilai 0 hingga 65535 . Input register bersifat *read only* bagi master. Data ini disimpan di register 30001 sampai 39999

d) Holding Register

Holding register digunakan untuk menyimpan nilai dgn range 0~65535. Register ini mempunyai alamat register 40001 sampai 49999.

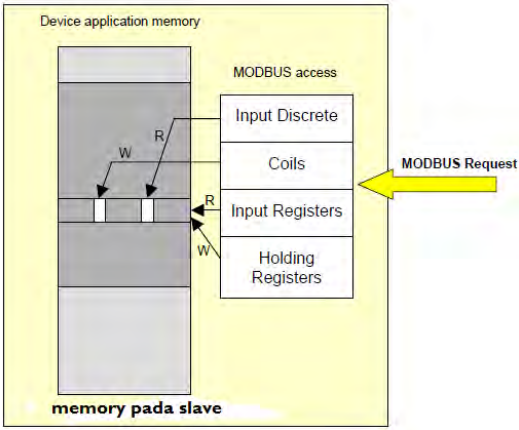
2.1.3 Pengorganisasian data modbus

Gambar dibawah ini memperlihatkan pengorganisasian data pada sebuah slave modbus yang masing masing mempunyai blok terpisah antara coil dan register.



Gambar 2.2 Memory dengan dengan coil dan register terpisah

Kemudian gambar dibawah ini memperlihatkan pengorganisasian data pada sebuah *slave* modbus yang hanya mempunayi satu blok untuk *coil* dan register.



Gambar 2.3 Memory dengan dengan satu blok untuk *coil* dan *register*

2.1.4 Frame Data Modbus

Master atau slave berkomunikasi dengan cara mengirim sebuah *frame* permintaan atau respon. Format *frame* ini digunakan untuk memastikan bahwa data yang terkirim lengkap tanpa kecacatan. Sehingga komunikasi terhindar dari kesalahan dalam penerimaan data.

Gambar 2.4 Frame Modbus

Frame modbus terdiri dari:

a) Alamat *slave*

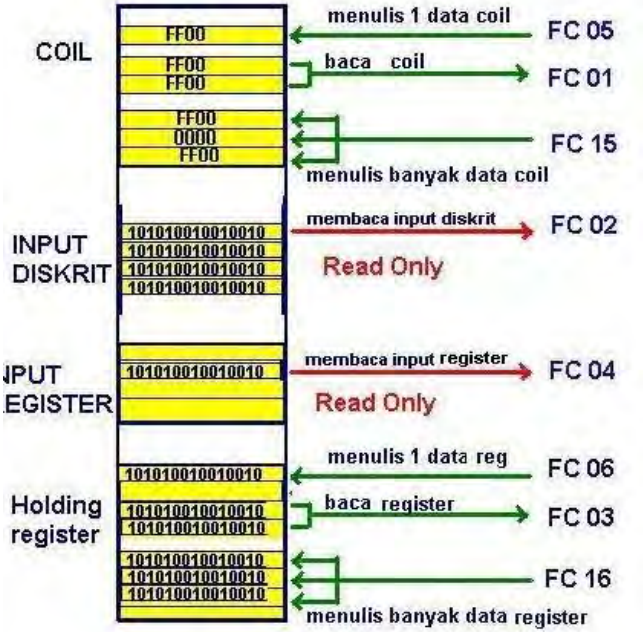
Byte pertama sebagai Alamat *slave* terdiri dari 1 byte (0~255). Namun alamat slave yang dapat diakses hanya 1 hingga 247. Alamat 0 ditujukan untuk perangkat master.

b) *Function Code*

Byte kedua berupa Function Code. Yaitu perintah fungsi akses data dari master yang harus dilakukan oleh *slave*. Berikut ini tabel kode perintah *function code*:

Tabel 2.2 *Function Code Modbus*

Data	Read 1 Data	Write 1 Data	Write Multiple Data	No. Awal Register
Coil	FC01	FC05	FC15	00001
Input Digital	FC02			10001
Input Register	FC04			30001
Holding Register	FC03	FC06	FC16	40001



Gambar 2.5 Ilustrasi register map dan *function code*

c) Byte Data

Jumlah Byte Data bervariasi tergantung jumlah data yg akan di tuliskan ke slave. Byte data berisi alamat register dan data yang akan ditulis.

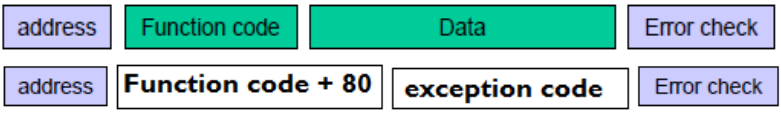
d) *Error check* (CRC)

CRC (*Cyclic Redundancy Check)* adalah algoritma untuk memastikan integritas data dan mengecek kesalahan pada suatu data yang akan ditransmisikan atau disimpan.

Data yang hendak ditransmisikan atau disimpan ke sebuah media penyimpanan rentan sekali mengalami kesalahan, seperti halnya noise yang terjadi selama proses transmisi atau memang ada kerusakan perangkat keras. Untuk memastikan integritas data yang hendak ditransmisikan atau disimpan, CRC dapat digunakan. CRC bekerja secara sederhana, yakni dengan menggunakan perhitungan matematika terhadap sebuah bilangan yang disebut sebagai Checksum, yang dibuat berdasarkan total bit yang hendak ditransmisikan atau yang hendak disimpan.

e) Respon MODBUS Exception

Respon *exeception* adalah respon dari *slave* ketika terjadi keadaan tidak normal/*error*. *Slave* menerima *query* , tetapi *Slave* tidak dapat menangani perintah tersebut, Slave akan mengirimkan sebuah respon *exception*. *frame* respon jika terjadi kesalahan berbeda dgn frame dlm keadaan normal.



Gambar 2.6 Perbedaan frame Modbus normal dan saat terjadi exception

Bila terdapat kesalahan pada pengiriman data, maka *slave* akan mengirimkan informasi kepada master dengan pesan *exception frame.* Berikut ini adalah daftar exception code yang terjadi bila terdapat kesalahan dalam pengiriman data :

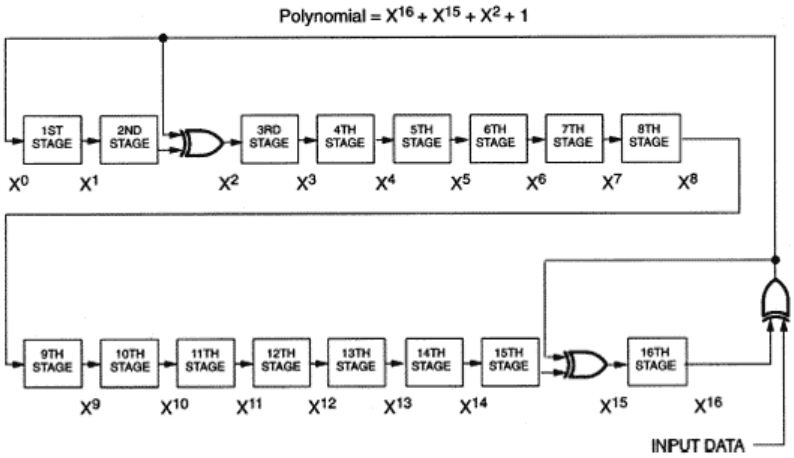
Tabel 2.3 Exception Code

Exception Code	Nama exception	Keterangan
01	*ILLEGAL FUNCTION*	Function code salah
02	*ILLEGAL DATA ADDRESS*	Alamat register salah atau tidak tersedia.
03	*ILLEGAL DATA VALUE*	Mengandung nilai data yang tidak diizinkan untuk slave.
04	*SLAVE DEVICE FAILURE*	Slave gagal melaksanakan perintah master
05	*ACKNOWLEDGMENT*	Pemberitahuan ke master bahwa pelaksanaan perintah akan memakan waktu yg lama, sehingga mengalami time out
06	*SLAVE DEVICE BUSSY*	Slave sidang sibuk

2.1.5 *Cyclic Redundancy Check* (CRC)

Data yang hendak ditransmisikan atau disimpan ke sebuah media penyimpanan rentan sekali mengalami kesalahan, seperti halnya noise yang terjadi selama proses transmisi atau memang ada kerusakan perangkat keras. Untuk memastikan integritas data yang hendak ditransmisikan atau disimpan, CRC dapat digunakan. CRC bekerja secara sederhana, yakni dengan menggunakan perhitungan matematika terhadap sebuah bilangan yang disebut sebagai Checksum, yang dibuat berdasarkan total bit yang hendak ditransmisikan atau yang hendak disimpan.

Dalam transmisi jaringan, khususnya dalam jaringan berbasis teknologi Ethernet, checksum akan dihitung terhadap setiap frame yang hendak ditransmisikan dan ditambahkan ke dalam frame tersebut sebagai informasi dalam header atau trailer. Penerima frame tersebut akan menghitung kembali apakah frame yang ia terima benar-benar tanpa kerusakan, dengan membandingkan nilai frame yang dihitung dengan nilai frame yang terdapat dalam header frame. Jika dua nilai tersebut berbeda, maka frame tersebut telah berubah dan harus dikirimkan ulang.

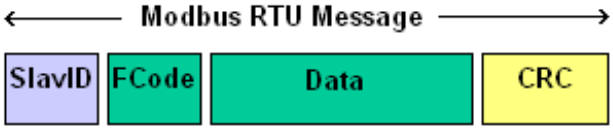


Gambar 2.7 Algoritma Polinomial CRC 16-bit

CRC didesain sedemikian rupa untuk memastikan integritas data terhadap degradasi yang bersifat acak dikarenakan noise atau sumber lainnya (kerusakan media dan lain-lain).

2.1.6 Modbus RTU

Protokol komunikasi Modbus RTU digunakan menggunakan komunikasi serial RS-485 dan menggunakan representasi nilai data biner yang dipadatkan sebagai protokol komunikasi. Format RTU mengikuti request perintah / transfer data dengan cyclic redundancy check checksum sebagai mekanisme pemeriksaan kesalahan (error-check) untuk memastikan keandalan data. Modbus RTU adalah implementasi yang paling umum dari semua versi Modbus yang ada. Sebuah pesan Modbus RTU harus dikirimkan secara terus menerus tanpa jeda antar-karakter. Setiap pesan Modbus dibingkai atau dipisahkan oleh periode-periode saat idle (tanpa komunikasi atau Port komunikasi ditutup atau OFF). Komunikasi via Modbus RTU sering dipakai dalam sistem monitoring skala kecil dimana sensor-sensor dan komputer HMI letaknya tidak sangat jauh.



Gambar 2.8 Frame data Modbus RTU

2.1.7 Modbus TCP

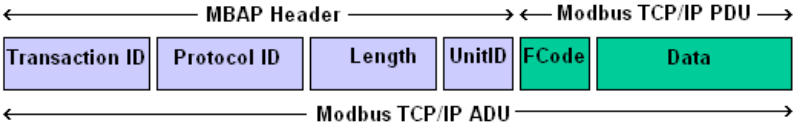
Transmission Control Protocol (TCP) dan Internet Protocol (IP) merupakan protokol yang digunakan secara bersamaan dan digunakan sebagai protokol transport untuk internet. Ketika informasi modbus dikirim menggunakan protokol ini, data yang akan diteruskan ke TCP mana informasi tambahan terpasang dan diberikan kepada IP. IP kemudian menempatkan data dalam paket (atau datagram) dan kemudian dikirim.

TCP harus membuat sambungan sebelum mentransfer data, karena merupakan protokol berbasis koneksi. Master (atau Client di Modbus TCP) menetapkan koneksi dengan Slave (atau Server). Server menunggu untuk koneksi masuk dari klien. Setelah sambungan dibuat, Server kemudian merespon permintaan dari klien sampai klien menutup koneksi.

Modbus TCP/IP lebih cepat dalam melakukan transfer data dibanding dengan Modbus RTU. Pada aplikasi sistem SCADA atau pun Automation yang kompleks dimana digunakan perangkat IED dalam jumlah yang banyak dan beraneka ragam atau dimana tingkat traffic transfer data yang padat, lebih disarankan menggunakan Modbus TCP/IP untuk mencapai tingkat real-time yang lebih tinggi. Namun tentu saja

perangkat IED dengan Port TCP/IP itu sendiri harganya relatif lebih mahal dibanding dengan Modbus RTU yang hanya menggunakan komunikasi Port RS-485.

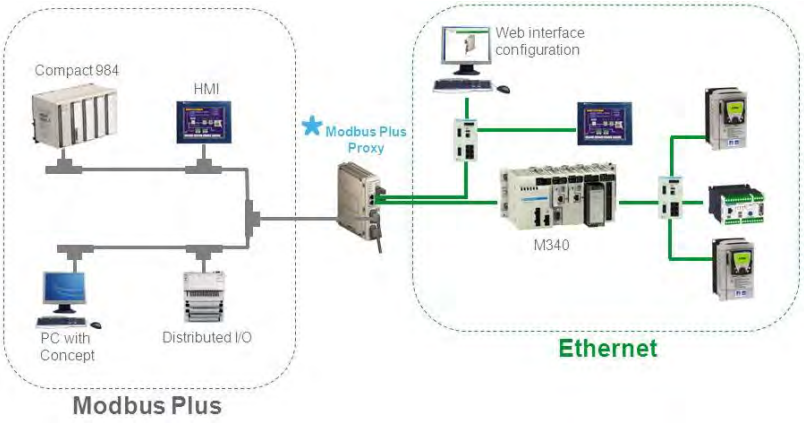
Perbedaan mendasar Modbus TCP dengan modbus RTU salah satunya adalah pada frame data yang dikirimkan. Berikut ini adalah frame data pada Modbus TCP :



Gambar 2.9 Frame Data Modbus TCP

2.1.8 Modbus Plus

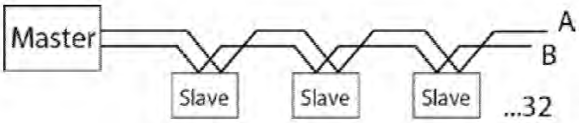
Modbus Plus (Modbus+ atau MB+) juga ada dan merupakan versi ekstensi dari semua versi Modbus, namun hanya eksklusif untuk SCHNEIDER ELECTRIC saja. Modbus ini membutuhkan co-prosesor khusus untuk menangani rotasi token secara cepat seperti HDLC. Modbus jenis ini menggunakan kabel twisted pair pada kecepatan 1 Mbit/s dan termasuk trafo isolasi di setiap node. Antarmuka khusus diperlukan sebagai penghubung Modbus Plus ke komputer, biasanya menggunakan card ISA (SA85), bus PCI atau PCMCIA yang khusus dibuat untuk MB+.



Gambar 2.10 Contoh Jaringan Modbus Plus (MB+)

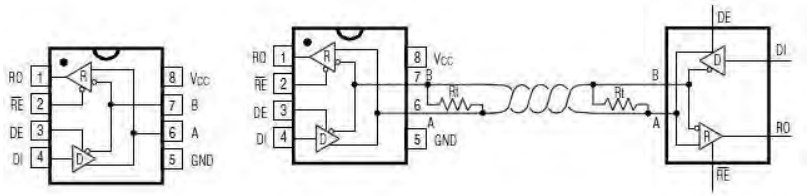
2.2 Komunikasi RS-485

RS485 adalah teknik komunikasi data serial yang dikembangkan di tahun 1983 dimana dengan teknik ini, komunikasi data dapat dilakukan pada jarak yang cukup jauh yaitu 1,2 Km. Berbeda dengan komunikasi serial RS232 yang mampu berhubungan secara one to one, maka komunikasi RS485 selain dapat digunakan untuk komunikasi multidrop yaitu berhubungan secara one to many dengan jarak yang jauh teknik ini juga dapat digunakan untuk menghubungkan 32 unit beban sekaligus hanya dengan menggunakan dua buah kabel saja tanpa memerlukan referensi ground yang sama antara unit yang satu dengan unit lainnya.



Gambar 2.11 Topologi RS-485

Bus RS485 adalah mode transmisi balanced differential. Bus ini hanya mempunyai dua sinyal, A dan B dengan perbedaan tegangan antara keduanya. Karena line A sebagai referensi terhadap B maka sinyal akan high bila mendapat input low demikian pula sebaliknya. Pada komunikasi RS485, semua peralatan elektronik berada pada posisi penerima hingga salah satu memerlukan untuk mengirimkan data, maka peralatan tersebut akan berpindah ke mode pengirim, mengirimkan data dan kembali ke mode penerima. Setiap kali peralatan elektronik tersebut hendak mengirimkan data, maka terlebih dahulu harus diperiksa, apakah jalur yang akan digunakan sebagai media pengiriman data tersebut tidak sibuk. Apabila jalur masih sibuk, maka peralatan tersebut harus menunggu hingga jalur sepi.



Gambar 2.12 Half Duplex dengan IC MAX485

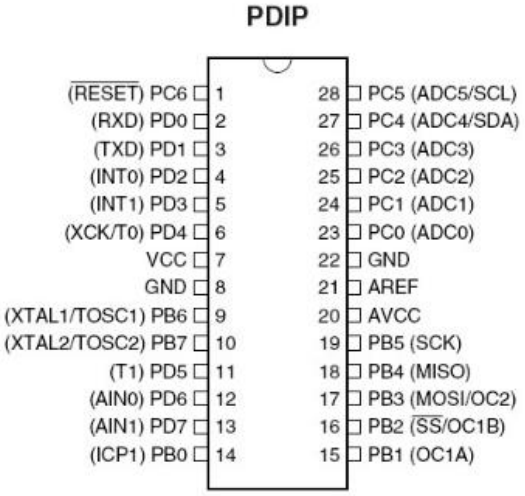
Agar data yang dikirimkan hanya sampai ke peralatan elektronik yang dituju, misalkan ke salah satu Slave, maka terlebih dahulu pengiriman tersebut diawali dengan Slave ID dan dilanjutkan dengan data yang dikirimkan. Peralatan elektronik yang lain akan menerima data tersebut, namun bila data yang diterima tidak mempunyai ID yang sama dengan Slave ID yang dikirimkan, maka peralatan tersebut harus menolak atau mengabaikan data tersebut. Namun bila Slave ID yang dikirimkan sesuai dengan ID dari peralatan elektronik yang menerima, maka data selanjutnya akan diambil untuk diproses lebih lanjut.

2.3 Mikrokontroler

Mikrokontroler berfungsi untuk membaca input dan menampilkan data melalui display 7 segment. Input dan output display tersebut dibaca dan dikendalikan oleh master melalui komunikasi Modbus RTU.

2.3.1 ATmega8

ATMega8 merupakan mikrokontroler keluarga AVR 8bit, memiliki 3 buah PORT utama yaitu PORTB, PORTC, dan PORTD dengan total pin input/output sebanyak 23 pin. PORT tersebut dapat difungsikan sebagai input/output digital atau difungsikan sebagai periperial lainnya.



Gambar 2.13 Atmega8

a) PORTB

PORTB merupakan jalur data 8bit yang dapat difungsikan sebagai input/output. Selain itu PORTB juga dapat memiliki fungsi alternatif seperti yang tertera pada tabel di bawah ini.

Tabel 2.4 Fungsi Alternatif PORTB

Port Pin	Alternate Function
PB7	XTAL2 (Chip Clock Oscillator pin 2) TOSC2 (Timer Oscillator pin 2)
PB6	XTAL1 (Chip Clock Oscillator pin 1) TOSC1 (Timer Oscillator pin 1)
PB5	SCK (SPI Bus Master Clock Input)
PB4	MISO (SPI Bus Master Output/Slave Output)
PB3	MOSI (SPI Bus Master Input/lave Input) OC2 (Timer/Counter Output Compare Match Output)
PB2	SS (SPI Bus Master Slave Select) OC1B (Timer/Counter1 Output Compare Match B Output)
PB1	OC1A (Timer/Counter1 Output Compare Match A Output)
PB0	ICP1 (Timer/Counter1 Input Capture Pin)

b) PORTC

PORTC merupakan jalur data 7 bit yang dapat difungsikan sebagai input/output digital. Fungsi alternatif PORTC antara lain sebagai berikut.

Tabel 2.5 Fungsi Alternatif PORTC

Port Pin	Alternate Function
PC6	Reset pin
PC5	ADC 5 (Analog to Digital pin 5) / SCL pin
PC4	ADC 4 (Analog to Digital pin 4) / SDA pin
PC3	ADC 3 (Analog to Digital pin 3)
PC2	ADC 2 (Analog to Digital pin 2)
PC1	ADC 1 (Analog to Digital pin 1)
PC0	ADC 0 (Analog to Digital pin 0)

c) PORTD

PORTD merupakan jalur data 8bit yang masing-masing pinnya juga dapat difungsikan sebagai input/output. Sama seperti PORTB dan PORTC, PORTD juga memiliki fungsi alternatif seperti terlihat pada gambar dibawah ini.

Tabel 2.6 Fungsi Alternatif PORTD

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input)
PD6	AIN0 (Analog Cmparator Positive Input)
PD5	T1 (Timer/Counter 1 External Counter Input)
PD4	XCK (USART External Cock Input/Output) T0 (Timer/Counter 0 External Counter Input)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

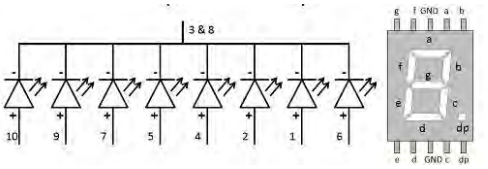
2.3.2 7 Segment

7 segment digunakan untuk menampilkan nomor urut yang sedang berlangsung di tiap slave. 7 segment merupakan komponen elektronika yang dapat menampilkan angka desimal maupun bilangan hexadesimal melalui kombinasi-kombinasi segmennya. Setiap segmen dikendalikan secara ON dan OFF untuk menampilkan karakter yang diinginkan. Pada dasarnya 7 Segment dibagi menjadi 2 jenis. Yaitu common anoda dan common katoda.

Gambar 2.14 Ilustrasi 7-Segment yang digunakan

a) Tipe 7 segment common cathode (Katoda)

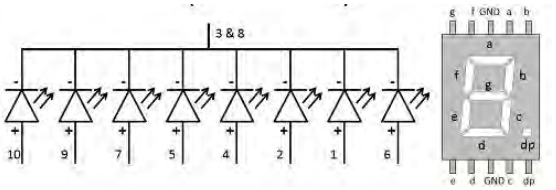
Pada LED 7 Segmen jenis Common Cathode (Katoda), Kaki Katoda pada semua segmen LED adalah terhubung menjadi 1 Pin, sedangkan Kaki Anoda akan menjadi Input untuk masing-masing Segmen LED. Kaki Katoda yang terhubung menjadi 1 Pin ini merupakan Terminal Negatif (-) atau Ground sedangkan Signal Kendali (Control Signal) akan diberikan kepada masing-masing Kaki Anoda Segmen LED.



Gambar 2.15 7-Segment Common Cathode

b) Tipe 7 segment common anoda

Pada LED 7 Segmen jenis Common Anode (Anoda), Kaki Anoda pada semua segmen LED adalah terhubung menjadi 1 Pin, sedangkan kaki Katoda akan menjadi Input untuk masing-masing Segmen LED. Kaki Anoda yang terhubung menjadi 1 Pin ini akan diberikan Tegangan Positif (+) dan Signal Kendali (control signal) akan diberikan kepada masing-masing Kaki Katoda Segmen LED.



Gambar 2.16 7 Segment Common Anode

2.3.3 IC 74HC595

IC ini digunakan untuk mengaktifkan segmen-segmen LED pada 7 segment untuk menampilkan data nomor urut antrian. 74HC595 adalah komponen yang memiliki fungsi sebagai shift register serial to parallel 8-bit, yaitu mengubah input berupa data serial ke bentuk output paralel 8bit. Komponen ini banyak dimanfaatkan sebagai ekspansi port pada

mikrokontroler. Dengan 3 buah pin (data, clock, dan latch) pengguna dapat mengontrol 8 pin, hingga ratusan pin sesuai jumlah 74HC595 yang digunakan. Macam-macam pin pada IC ini adalah OE (Output Enable), MR (Reset), DS (Data), SH_CP (Clock), ST_CP (Latch) dan Q0 s/d Q7.

Gambar 2.17 IC 74HC595

Tabel 2.7 Pin pada 74HC595

Nomor Pin	Nama Pin	Kegunaan
Pin 1-7, 15	Q0 - Q7	Pin Output
Pin 8	GND	Ground, Vss
Pin 9	Q7’	Output Serial
Pin 10	MR	Master Reclear, Active low
Pin 11	SH_CP	Shift register clock pin
Pin 12	ST_CP	Storage Register latch pin
Pin 13	OE	Ouutput Enable
Pin 14	DS	Serial Data Input
Pin 16	Vcc	TEgangan Sumber Positif

2.4 Sistem Antrian

Antrian merupakan bagian dalam sebuah proses atau pelayanan. Dalam hal mengantri, waktu merupakan komponen atau aspek yang sangat penting dan berharga. Oleh karena itu sedapat mungkin sistem yang ada dapat mereduksi penggunaan waktu yang berlebihan, sehingga tercapainya sistem yang efektif dan efisien dalam hal peggunaan waktu tersebut. Waktu mengantri juga menjadi komponen yang sangat penting, hal ini dikarenakan berhubungan dengan peningkatan kualitas dari pelayanan itu sendiri.

Menurut Siagian (1987), antrian ialah suatu garis tunggu dari nasabah (satuan) yang memerlukan layanan dari satu atau lebih pelayan (fasilitas layanan). Pada umumnya, sistem antrian dapat diklasifikasikan menjadi system yang berbeda – beda di mana teori antrian dan simulasi

sering diterapkan secara luas. Klasifikasi menurut Hillier dan Lieberman adalah sebagai berikut

a) Sistem pelayanan komersial
b) Sistem pelayanan bisnis – industri
c) Sistem pelayanan transportasi
d) Sistem pelayanan social

Sistem pelayanan komersial merupakan aplikasi yang sangat luas dari model – model antrian, seperti restoran, kafetaria, toko, salon, butik, supermarket, dan sebagainya. Sistem pelayanan bisnis – industri mencakup lini produksi, sistem material – handling, sistem pergudangan, dan sistem – sistem informasi komputer. Sistem pelayanan sosial merupakan sistem – sistem pelayanan yang dikelola oleh kantor – kantor dan jawatan – jawatan lokal maupun nasional, seperti kantor registrasi SIM dan STNK, kantor pos, rumah sakit, puskesmas, dan lain – lain (Subagyo, 2000). Ada tiga komponen dalam sistem antrian yaitu :

a) Populasi

Populasi yang akan Dilayani (calling population) Setiap masalah antrian melibatkan kedatangan, misalnya orang, mobil, panggilan telepon untuk dilayani, dan lain – lain. Unsur ini sering dinamakan proses input. Proses input meliputi sumber kedatangan atau biasa dinamakan calling population, dan cara terjadinya kedatangan yang umumnya merupakan variabel acak.

Karakteristik dari populasi yang akan dilayani (calling population) dapat dilihat menurut ukurannya, pola kedatangan, serta perilaku dari populasi yang akan dilayani. Menurut ukurannya, populasi yang akan dilayani bisa terbatas (finite) bisa juga tidak terbatas (infinite). Sebagai contoh jumlah mahasiswa yang antri untuk registrasi di sebuah perguruan tinggi sudah diketahui jumlahnya (finite), sedangkan jumlah nasabah bank yang antri untuk setor, menarik tabungan, maupun membuka rekening baru, bisa tak terbatas (infinte).

b) Sistem Pelayanan Antrian

Sistem Pelayanan Antrian meliputi beberapa hal yakni garis antrian/ baris tunggu dan ketersediaan fasilitas. Faktor-faktor yang

terkait dengan garis antrian meliputi panjang antrian, jumlah baris antrian dan disiplin antrian. Jumlah antrian dalam sistem antrian dikelompokkan menjadi dua yakni antrian tunggal. Artinya hanya ada satu fasilitas layanan untuk melayani antrian. 2) Antrian berganda/ multi. Artinya ada beberapa fasilitas layanan di depan baris antrian.

Disiplin antrian dikelompokkan menjadi dua, yaitu preemptive dan non preemptive. Disiplin preemptive menggambarkan situasi dimana pelayan sedang melayani seseorang, kemudian beralih melayani orang yang diprioritaskan meskipun belum selesai melayani orang sebelumnya. Sementara disiplin non preemptive menggambarkan situasi dimana pelayan akan menyelesaikan pelayanannya baru kemudian beralih melayani orang yang diprioritaskan. Sedangkan disiplin first come first serve menggambarkan bahwa orang yang lebih dahulu datang akan dilayani terlebih dahulu. Dalam kenyataannya sering dijumpai kombinasi dari tersebut. Yaitu prioritas dan first come first serve. Sebagai contoh, para pembeli yang akan melakukan pembayaran di kasir untuk pembelian kurang dari sepuluh jenis barang (dengan keranjang) di super market disediakan counter tersendiri.

c) Kondisi pelanggan saat keluar sistem

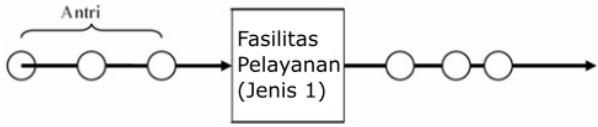
Setelah pelanggan dilayani, ada dua kemungkinan kondisi pelanggan itu keluar sistem: 1) pelanggan mungkin kembali ke populasi sumber dan mengantri lagi, atau 2) pelanggan hanya kemungkinan kecil untuk mendapat pelayanan ulang.

Gambar 2.18 Ilustrasi sistem antrian

Ada 4 model struktur antrian dasar yang umum terjadi dalam seluruh sistem antrian, yaitu :

a) Single Channel - Single Phase

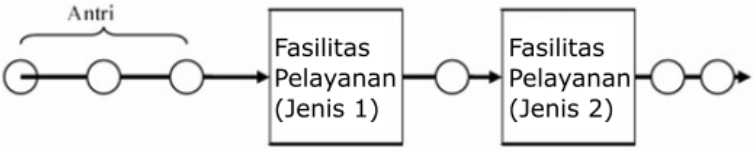
Single Channel berarti hanya ada satu jalur yang memasuki system pelayanan atau ada satu fasilitas pelayanan. Single Phase berarti hanya ada satu pelayanan.



Gambar 2.19 Struktur Antrian Single Channel - Single Phase

b) Single Phase - Multi Channel

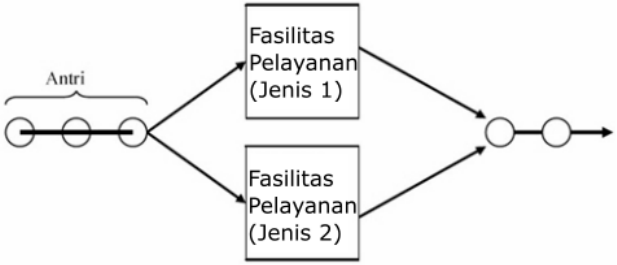
Istilah Multi Phase menunjukkan ada dua atau lebih pelayanan yang dilaksanakan secara berurutan (dalam phasephase). Sebagai contoh : pencucian mobil.



Gambar 2.20 Struktur Antrian Single Phase - Multi Channel

c) Multi Channel - Single Phase

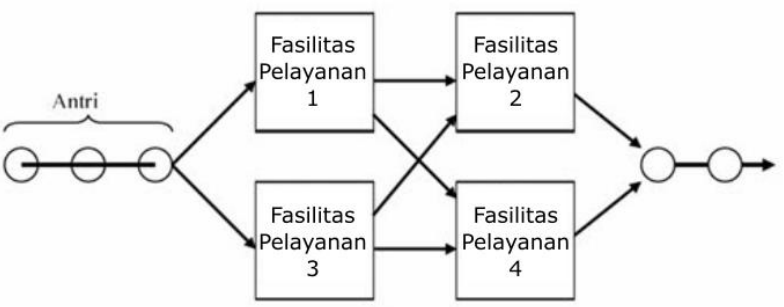
Sistem Multi Channel – Single Phase terjadi kapan saja di mana ada dua atau lebih fasilitas pelayanan dialiri oleh antrian tunggal, sebagai contoh model ini adalah antrian pada teller sebuah bank.



Gambar 2.21 Struktur Antrian Multi Channel - Single Phase

d) Multi Channel – Multi Phase

Sistem Multi Channel – Multi Phase Sebagai contoh, herregistrasi para mahasiswa di universitas, pelayanan kepada pasien di rumah sakit mulai dari pendaftaran, diagnosa, penyembuhan sampai pembayaran. Setiap sistem – sistem ini mempunyai beberapa fasilitas pelayanan pada setiap tahapnya (Subagyo, 2000).



Gambar 2.22 Struktur Antrian Multi Channel – Multi Phase

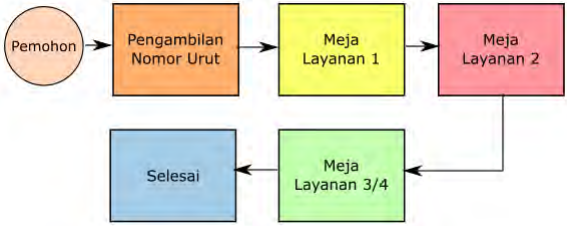
--Halaman ini sengaja dikosongkan--

BAB III
PERANCANGAN DAN REALISASI ALAT

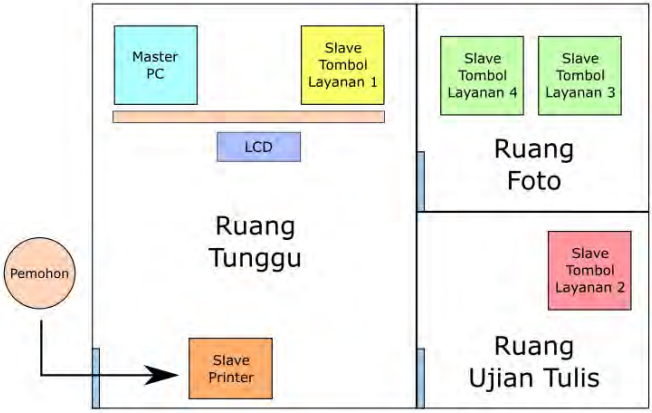
Pada bab ini menjelaskan tentang perancangan dan implementasi sistem meliputi arsitektur sistem, perancangan *hardware*, *software*, dan realisasi alat.

3.1 Perancangan Sistem

Sistem terdiri dari beberapa slave berupa mikrokontroler dengan input tombol dan display 7-segment yang dapat dibaca dan dikendalikan oleh sebuah PC sebagai master. Berikut ini adalah ilustrasi diagram sistem yang dibuat :



Gambar 3.1 Skema Urutan Sistem Antrian



Gambar 3.2 Ilustrasi Ruang Antri

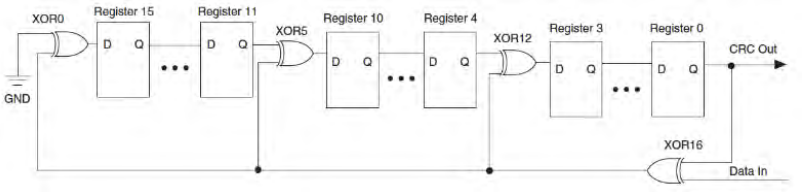
Slave berfungsi untuk membaca status tombol dan menyimpannya di dalam *holding register* yang kemudian akan dikirim jika terjadi request oleh master, kemudian master akan mengisi nilai *holding register* untuk menampilkan angka pada *Display 7-segment* sesuai dengan nomor urut yang berlangsung. Kemudian terdapat sebuah *thermal printer* dengan input tombol untuk mencetak nomor urut untuk pemohon. Data nomor urut tersebut dibaca oleh master untuk disimpan sebagai nomor urut selanjutnya.

3.2 Perancangan *Software*

Perancangan *software* meliputi interfacing yang digunakan untuk menampilkan tampilan antrian utama dan menjalankan peran sebagai master untuk mengontrol dan mengkoleksi data dari slave.

3.2.1 Implementasi Algoritma Cyclic Redundancy Check (CRC)

Untuk dapat memenuhi syarat komunikasi Modbus, master maupun slave harus memiliki kemampuan atau fungsi untuk menerjemahkan kode CRC.



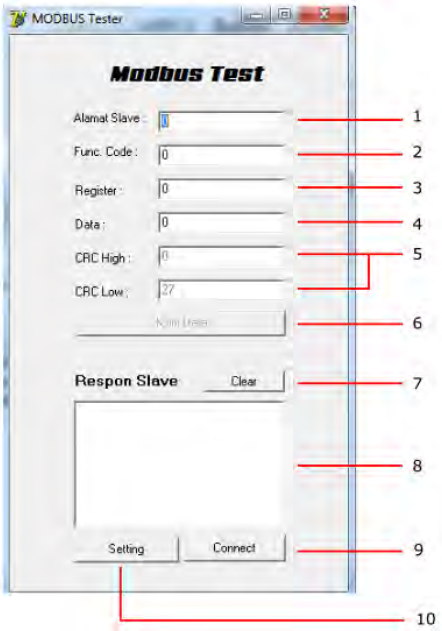
Gambar 3.3 Rangkaian logika CRC 16-bit

Berdasarkan rangkaian logika tersebut, dibuatlah listing program CRC dengan menggunakan bahasa pemrograman Pascal sebagai berikut ini:

```
function CRC16(Buffer:String; len: integer):Cardinal;
var
  i,j, CRCLSB, CRCFull: Integer;
begin
  CRCFull := $FFFF;

  for i := 1 to len do
  begin
    CRCFull := CRCFull xor ord(buffer[i]);
      for j := 0 to 7 do
      begin
        CRCLSB := CRCFull and $0001;
        CRCFull := (CRCFull shr 1) and $7FFF;

        if  CRCLSB <> 0 then CRCFull := CRCFull xor $A001;
      end;
  end;

  Result := CRCFull;
end;
```

3.2.3 Modbus Tester

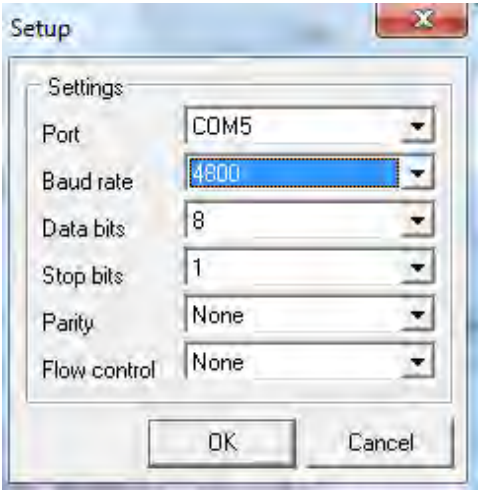
Interface ini dibuat untuk simulasi pengiriman data frame Modbus. Tujuan dari simulasi ini adalah untuk mengecek apakah komunikasi Modbus pada perangkat slave yang terlah dibuat berjalan dengan baik atau tidak, agar dapat melakukan debugging dengan lebih mudah. Berikut ini adalah tampilan dari interface Modbus Tester sederhana yang telah dibuat :



Gambar 3.4 Software Interface Modbus Tester

Berikut ini fitur-fitur yang terdapat pada software interface Modbus Tester :

a) Alamat Slave yang dituju
b) Function code yang digunakan
c) Register yang ingin diakses
d) Isi data yang akan dikirimkan
e) CRC High dan Low otomatis dihitung dan terisi sesuai dengan input yang dimasukkan.
f) Kirim frame data
g) Bersihkan text box respon slave
h) Mulai koneksi RS485
i) Setting baudrate dan com port



Gambar 3.5 Window Setting Com Port dan Baudrate

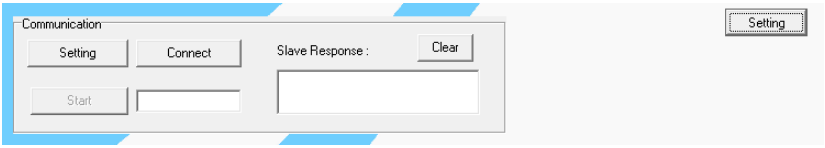
Input yang dimasukkan melalui Modbus Tester ini adalah data integer yang secara otomatis akan dirubah dan dikirimkan dalam bentuk karakter, dan sebaliknya. Apabila koneksi berjalan dengan baik, maka akan terdapat balasan dari slave.Software ini berguna untuk mengecek apakah koneksi berjalan dengan baik di tiap *slave*. Apabila tidak terdapat respon pada *slave* maka dapat melakukan *debugging* pada perangkat *slave*.

3.2.4 Interface Utama

Merupakan tampilan utama yang menampilkan nomor urut antrian pelayanan yang ditampilkan di ruang antri. Interface ini juga berperan sebagai master yang mengolah data dari slave.

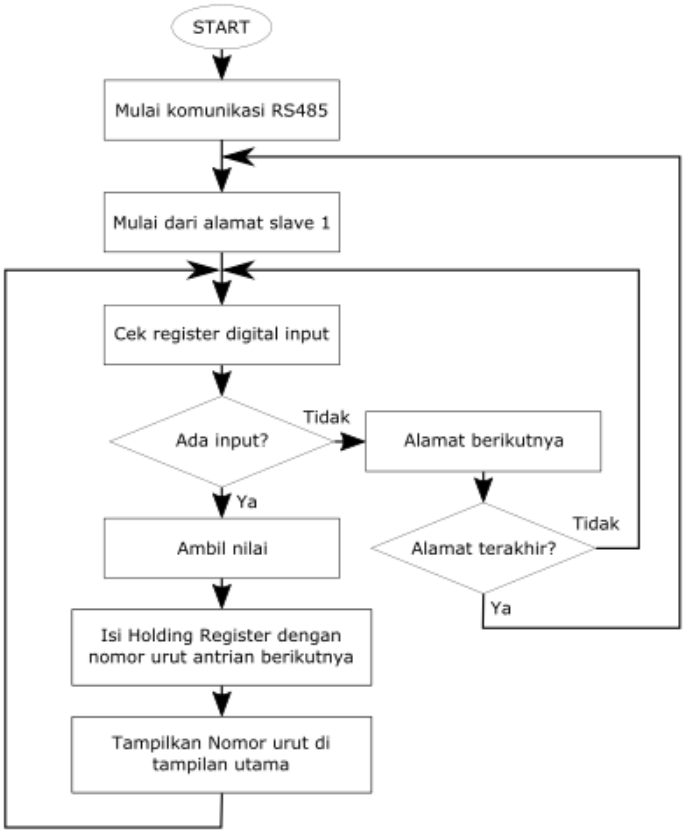
Gambar 3.6 Tampilan Interface Utama

Dalam interface ini terdapat 4 buah urutan angka yang ditampilkan. Terdiri dari pembayaran Loket BRI, Ujian Teori, dan Ruang Foto SIM. Terdapat media player yang dapat digunakan untuk menampilkan iklan maupun video edukasi. Selain itu untuk proses debugging, terdapat tombol setting untuk melihat proses pooling dari master ke slave.



Gambar 3.7 Setting koneksi dan display data respon dari slave

Cara kerja dari interface utama ini adalah menscan perangkat slave dan mengecek apakah terdapat input digital. Jika 'Ya', maka akan dilakukan aksi yaitu mengisi holding register berupa urutan antrian selanjutnya dan meneruskan proses scanning. Berikut ini flowchart dari program interface utama:



Gambar 3.8 Flowchart Program Interface Utama

3.3 Perancangan *Hardware*

Karena master berupa PC, perancangan *hardware* sistem antrian yang dilakukan hanya pada bagian slave. Terdiri dari kontroler, tombol, printer thermal, dan display.

3.3.1 Rangkaian Kontroler Slave

Slave terdiri dati tombol dan display dengan ATmega8 sebagai kontroler utama. Port yang digunakan terdiri dari pin input tombol, socket

untuk konverter TTL to RS485, pin untuk output display, dan konektor ISP.

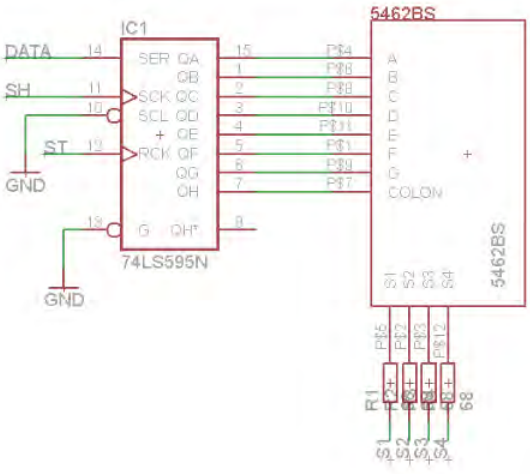
Untuk lebih lengkapnya mengenai penggunaan port ATmega8 pada kontroler slave, dapat melihat tabel berikut ini.

Tabel 3.1 Penggunaan Port ATmega8

No	Pin	Kegunaan
1	PD0	Pin receiver (Rx) ke data in konverter RS485
2	PD1	Pin transmitter (Tx) ke data out konverter RS485
3	PD2	Data Control (mengatur slave sebagai receiver atau transmitter)
4	PD3	Input untuk next button
5	PD4	Input untuk skip button
6	PB0	Scan pin 7-Segment (Satuan)
7	PB1	Scan pin 7-Segment (Puluhan)
8	PB2	Scan pin 7-Segment (Ratusan)
9	PB3	Scan pin 7-Segment (Ribuan)
10	PB4	Serial Data pin 74HC595
11	PB6	Storage pin 74HC595
12	PB7	Serial clock pin 74HC595

3.3.2 Perancangan Display 7-Segment

Display terdiri dari 4 buah 7-segment yang mewakili nilai ratusan, puluhan, dan satuan. Agar meminimalisir pengguaan port, digunakan IC shift register 74HC595 sebagai perantara scanning untuk 7-segment.



Gambar 3.9 Rangkaian Display 7-Segment

3.3.3 Rangkaian Konverter TTL to RS485

Untuk merubah logika TTL dari mikrokontroler menjadi mode transmisi balanced differential yang hanya mempunyai dua sinyal, A dan B. Pada komunikasi RS485, semua slave berada pada posisi penerima hingga master me-*request* data, maka slave yang dituju akan berpindah ke mode pengirim, mengirimkan data dan kembali ke mode penerima. Untuk mempermudah hal maka diperlukan konverter. Disini dipilih modul max485.

Gambar 3.10 Modul konverter Max485

3.3.4 Kontroler Printer Thermal

Printer Thermal digunakan sebagai pencetak nomor urut yang dibawa oleh peserta antrian sebagai tanda bukti. Pada printer ini terdapat tombol pilihan yang nantinya akan disimpan dan dikirim ke master untuk diurutkan sebagai nomor antrian selanjutnya.

Printer yang digunakan merupakan printer yang menggunakan komunikasi TTL sehingga dapat langsung dikendalikan melalui mikrokontroler ATmega8.

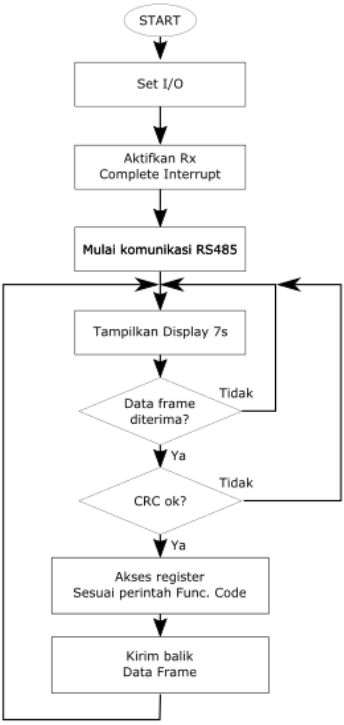
Gambar 3.11 Printer Thermal TTL

3.3.5 Perancangan Program Slave

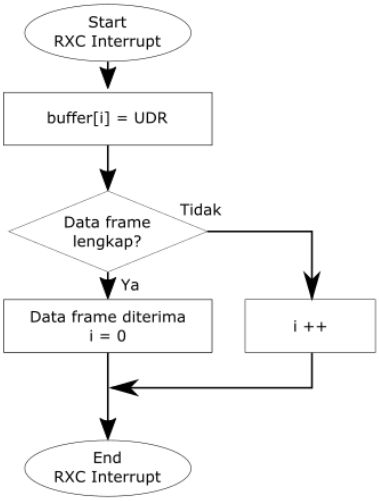
.Mikrokontroler harus mampu mengolah data frame yang diterima dan mengirimkan kembali informasi yang dimiliki ke perangkat master.

Pada saat pertamakali menyala, mikrokontroler akan menginisialisasi port I/O dan memulai komunikasi. Lalu menampilkan data ankga dari holding register ke display 7-segment. Saat data frame diterima dan nilai CRC benar, maka data frame akan di parsing dan diolah sesuai dengan perintah function code yang diberikan oleh master. Kemudian mengirim balik data yang diakses menuju master.

Rx interrupt digunakan agar proses penerimaan karakter tidak terganggu dengan looping program utama. Karena pada program terdapat delay untuk menampilkan display 7-segment dan sebagainya, maka dengan Rx interrupt karakter yang dikirim oleh master bisa diterima secara utuh. Berikut ini adalah flowchart sistem kerja dari perangkat slave :



Gambar 3.12 Flowchart Program Slave (Mikrokontroler)



Gambar 3.13 Flowchart Rx Interrupt

Lalu agar CRC dapat diimplementasikan pada perangkat slave (mikrokontroler) listing program disesuaikan menggunakan bahasa C seperti berikut ini :

```
unsigned long CRC(char message[], int panjangdata, int mode)
{
    unsigned char i,j;
    unsigned int CRCFull = 0xFFFF;
    unsigned int CRCH = 0xFF;
    unsigned int CRCL = 0xFF;
    unsigned  char CRCLSB;

    for (i = 0; i < (panjangdata) ; i++)
    {
        CRCFull = (CRCFull ^ message[i]);

        for ( j = 0; j < 8; j++)
        {
            CRCLSB = (CRCFull & 0x0001);    //ambil LSB
            CRCFull = ((CRCFull >> 1) & 0x7FFF);  // geser
```

```
            if (CRCLSB == 1)
            CRCFull = CRCFull ^ 0xA001;
        }
    }

    CRCH = ((CRCFull >> 8) & 0xFF);
    CRCL = (CRCFull & 0xFF);

    if(mode == 0)
    {
        if(CRCH == message[7] && CRCL == message[6]) return 1;
        else return 0;
    }
    else return CRCFull;
}
```

Dari fungsi tersebut, nilai balik yang dapat digunakan ada tiga. Pertama apabila mode = 0, nilai balik antara 1 atau 0 yang menandakan sama atau tidaknya CRC yang diterima. Jika mode = 1 maka nilai baliknya adalah hasil perhitungan CRC yang digunakan untuk mengisi frame data yang akan dikirimkan kembali.

--Halaman ini sengaja dikosongkan--

BAB IV
PENGUJIAN DAN ANALISA SISTEM

Bab ini menjelaskan tentang hasil pengujian dan analisa dari perancangan sistem yang telah dilakukan dan dijelaskan pada bab sebelumnya. Pengujian dari sistem antrian ini dilakukan menjadi dua tahap, yaitu pengujian hardware dan pengujian software. Pengujian hardware berupa pengujian register dan jarak komunikasi dengan panjang kabel tertentu. Pengujian software meliputi pengujian input, output, dan kinerja sistem komunikasi.

4.1 Pengujian Hardware

Pengujian dilakukan untuk mengetahui kondisi perangkat apakah dapat berjalan sesuai dengan kinerja yang dharapkan. Pengujian terdiri dari pengujian jarak, pengujian komunikasi, dan pengukuran daya.

4.1.1 Pengujian Register

Terdapat 3 buah register yang digunakan pada perangkat slave. 2 buah coil register dan 1 buah holding register. Pengujian ini dilakukan untuk mengecek tiap register pada slave.

a) Pengujian coil register, yaitu dengan request read coil di register 00001 dan 00002 yang mewakili tombol Next dan Skip. Berikut ini tabel perbandingan query dan response pada saat coil tidak aktif (0000H) menggunakan function code FC01 :

Tabel 4.1 Hasil pengujian pada Coil register (non aktif)

No. Slave	Data Yang dikirim	Data Diterima
1	01 01 01 00 00 109 202	01 01 01 00 00 109 202
2	02 01 01 00 00 109 249	02 01 01 00 00 109 249
3	03 01 01 00 00 108 40	03 01 01 00 00 108 40
4	04 01 01 00 00 109 159	04 01 01 00 00 109 159
5	01 01 02 00 00 157 202	01 01 02 00 00 157 202
6	02 01 02 00 00 157 249	02 01 02 00 00 157 249
7	03 01 02 00 00 156 40	03 01 02 00 00 156 40
8	04 01 02 00 00 157 159	04 01 01 00 00 157 159

b) Kemudian jika tombol ditekan, maka coil akan aktif. Coil dengan register 00001 mewakili tombol Next sedangkan register 00002 mewakili tombol skip. Pada register ini, untuk mengaktifkan sebuah relay (ON) nilai yang digunakan adalah FF00H atau 255 dan 0. Dibawah ini adalah tabel perbandingan query dan response pada saat coil aktif menggunakan function code FC01 :

Tabel 4.2 Hasil pengujian pada Coil register (aktif)

No. Slave	Data Yang dikirim	Data Diterima
1	01 01 01 00 00 109 202	01 01 01 255 00 44 58
2	02 01 01 00 00 109 249	02 01 01 255 00 44 09
3	03 01 01 00 00 108 40	03 01 01 255 00 45 216
4	04 01 01 00 00 109 159	04 01 01 255 00 44 111
5	01 01 02 00 00 157 202	01 01 02 255 00 220 58
6	02 01 02 00 00 157 249	02 01 02 255 00 220 09
7	03 01 02 00 00 156 40	03 01 02 255 00 221 216
8	04 01 02 00 00 157 159	04 01 02 255 00 220 111

c) Pengujian holding register, yaitu dengan mengirim data angka 1234 di register 40001. Jika data berhasil disampaikan maka query akan dikirimkan kembali dan 7-segment tampil sesuai data angka yang dikirim.

Tabel 4.3 Hasil pengujian pada Holding register

No. Slave	Data Yang dikirim	Data Diterima
1	01 06 01 42 10 90 151	01 06 01 42 10 90 151
2	02 06 01 42 10 90 164	02 06 01 42 10 90 164
3	03 06 01 42 10 91 117	03 06 01 42 10 91 117
4	04 06 01 42 10 90 194	04 06 01 42 10 90 194

4.1.2 Pengujian Jarak Komunikasi

Pada pengujian ini, akan digunakan kabel dengan panjang bervariasi untuk melihat respon perangkat terhadap jarak komunikasi. Berikut ini hasil pengujian yang telah dilakukan:

Tabel 4.4 Hasil pengujian jarak komunikasi

No.	Panjang Kabel	Data yang Dikirim	Data yang Diterima	Hasil
1	100m	01 06 01 42 10 90 151	01 06 01 42 10 90 151	OK
2	200m	02 06 01 62 25 68 113	02 06 01 62 25 68 113	OK
3	300m	01 06 01 35 83 12 97	01 06 01 35 83 12 97	OK
4	400m	03 06 01 16 225 20 96	03 06 01 16 225 20 96	OK
5	500m	04 06 01 17 38 85 213	04 06 01 17 38 85 213	OK

4.1.3 Pengujian Konsumsi Daya

Pengujian ini dilakukan untuk mengetahui spesifikasi rating daya seseluruhan yang akan digunakan pada sistem antrian terintegrasi. Pengukuran terdiri dari arus dan tegangan yang dibutuhkan pada tiap perangkat slave.

Tabel 4.5 Hasil pengukuran konsumsi daya

No.	Perangkat	Tegangan	Arus	Daya
1	4 buah Slave Tombol	5 Volt	0.175 A	0.7 Watt
2	Slave Printer (Print Aktif)	9 Volt	1.535 A	13.815 Watt
Total				14.515 Watt

4.2 Pengujian Software

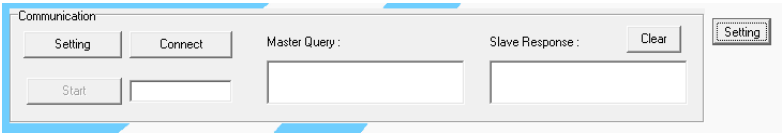
Dalam perancangan software antrian ini dilakukan pengujian sebagai verifikasi untuk mengetahui kemungkinan terjadinya kesalahan dan untuk memastikan fungsi-fungsi yang terdapat dalam modul-modul aplikasi tersebut berjalan dengan baik.

Gambar 4.1 Tampilan software

4.2.1 Tampilan Interface

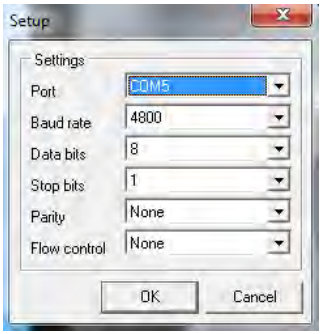
Software yang dibuat merupakan sebuah interface yang menjembatani antara device slave dengan PC sebagai master. Untuk dapat melakukan hal tersebut, software harus memiliki kemampuan untuk membaca port serial, mengolah data yang terbaca menjadi satu buah frame protokol modbus, kemudian mengurai frame tersebut untuk mendapatkan data yang dibutuhkan.

Berikut ini adalah tampilan untuk melihat data frame yang dikirim dan diterima oleh software interface master:



Gambar 4.2 Tampilan Komunikasi Data Master-Slave

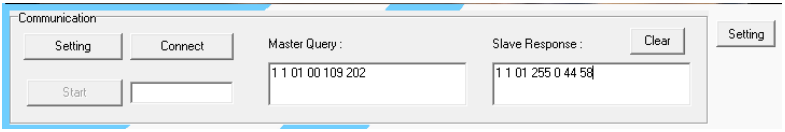
Selain data frame, pada tampilan tersebut terdapat beberapa fungsi diantaranya tombol ‘Connect’ untuk memulai koneksi, tombol ‘Clear’ untuk mengosongkan text area, dan tombol ‘Setting’ untuk mengatur properti komunikasi serial yang akan dijalankan.



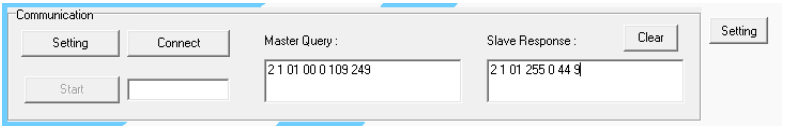
Gambar 4.3 Setup properti komunikasi

4.2.2 Pengujian Komunikasi

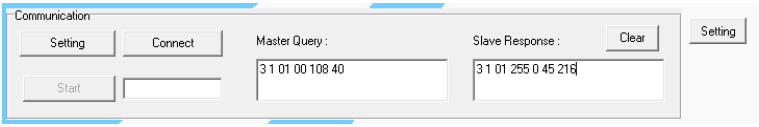
Pada bagian ini akan ditampilkan hasil pengujian koneksi master dengan slave melalui software interface yang telah dibuat. Antara master dan slave akan menjalankan protokol Modbus RTU dimana data yang dikirimkan berupa byte-byte karakter yang telah diatur oleh aturan data frame Modbus RTU. Bila data yang diterima valid, maka data akan diproses dan dikirim kembali ke master.



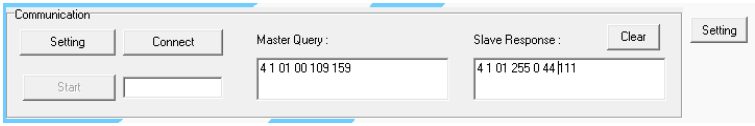
Gambar 4.4 *Software* membaca *Coil Register Slave* 1



Gambar 4.5 *Software* membaca *Coil Register Slave* 2



Gambar 4.6 *Software* membaca *Coil Register Slave* 3



Gambar 4.7 *Software* membaca *Coil Register Slave* 4

Ketika data respon di *coil register* bernilai FF00h, maka tombol dianggap aktif dan nomor antrian bertambah. Data nomor ini dikirim dan disimpan di *holding register* pada *slave* menggunakan function code FC06. Kemudian slave akan mencacah data frame yang dikirim dan jika function code FC06 terdeteksi, maka slave akan menulis data di holding register yang dituju. Setelah proses selesai, maka data frame tersebut dikembalikan ke master untuk menandakan bahwa proses penulisan register berhasil. Berikut ini adalah hasil uji coba penulisan di holding register pada slave :

Gambar 4.8 Software menulis Holding Register Slave 1

Gambar 4.9 Software menulis Holding Register Slave 2

Gambar 4.10 Software menulis Holding Register Slave 3

Gambar 4.11 Tampilan pada Software

4.3 Analisa

Berikut analisa yang diperoleh dari perbandingan hasil pengujian dengan perancangan pada bab sebelumnya antara lain:

1. Komponen ComPort yang digunakan di software interface tidak memiliki kemampuan untuk memisahkan data 3,5 karakter, sehingga harus menyesuaikan agar dapat membaca data sebagai satu kesatuan frame dengan cara menghirtung karakter yang masuk dan memasukkannya ke dalam char array memanfaatkan timer.

2. Terdapat beberapa perbedaan pada saat melakukan perngujian secara parsial (satu per satu) dengan pengujian secara bersamaan (semua slave terhubung ke dalam satu jaringan). Diantaranya adalah terkadang data tidak dapat terbaca langsung sehingga perlu perulangan untuk memastikan data telah benar-benar diproses.

3. Komunikasi dengan protokol Modbus RTU berhasil meskipun masih mengalami beberapa data error. Mikrokontroler mampu menerima dan mengolah data dengan baik, dan software interface juga mampu mengendalikan sistem yang dibuat.

LAMPIRAN A

A.1 *Listing* Program C

```
#include <stdlib.h>
#include <stdio.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/eeprom.h>

#ifndef F_CPU
#define F_CPU 8000000UL
#endif
#define BAUD 4800
#define BAUDRATE ((F_CPU)/(BAUD*16UL)-1)

#define    Tx                                    0
#define    Rx                                    1
#define next                                !(PIND & (1 << PD3))
#define skip                                !(PIND & (1 << PD4))
#define delay                               _delay_ms
#define lcd_clear               lcd_clrscr
#define tx_mode                   PORTD|=(1<<PD2)
#define rx_mode                 PORTD&=~(1<<PD2)
#define    dev_address                      1

char buffer[8], temp[10], temp2[10];
unsigned int indexbuf, i, no;
unsigned int reg0000[9]; //Coil Registers
unsigned long reg4000[9]; //Holding Registers
unsigned char * hexc;

void setDisp(unsigned int angka);
unsigned long CRC(char message[], int panjangdata, int mode);

void uart_putc(const uint8_t c)
{
        while (!(UCSRA & (1<<UDRE)));  // wait until ready to send
        UDR = (char)c;                  // send charakter
}

void uart_puts (const char *s)
```

```
{
        do
        {
            uart_putc(*s);                  //send one char of string
        }
        while(*s++);                        //next char of string until end of string sign
}

void uart_init()
{
        UBRRH = (BAUDRATE>>8);
    UBRRL = BAUDRATE;
    UCSRB|= (1<<TXEN)|(1<<RXEN)|(1 << RXCIE );
    UCSRC|= (1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);
}

unsigned char * uart_gets()
{
    unsigned char * x;
    while(!(UCSRA & (1<<RXC)));

    while(UDR != '\0')
    {
        *x = UDR;
        x++;
        while ( !(UCSRA & (1<<RXC)) );
    }
    *x = UDR;

     return x;
}

ISR(USART_RXC_vect)
{
        char c = UDR;
        if(indexbuf  <= 7)
        {
            buffer[indexbuf] = c;
            indexbuf++;
        }
}

int main()
```

```
{
	DDRB = 0xFF;
	PORTB = 0x00;
	DDRC = 0x00;
	PORTC = 0x00;
	DDRD |= (1<<PD2)|(1<<PD6)|(1<<PD7);
	PORTD |= (1<<PD3)|(1<<PD4);

	uart_init();
  sei();
  rx_mode;

	while(1)
	{
		setDisp(no);

		if(next && reg0000[1] == 0 && reg0000[2] == 0)
		{
			reg0000[1] = 1;
			for(int i=0; i<3; i++)
			{
				delay(100);
				setDisp(no);
			}
		}

		if(skip && reg0000[2] == 0 && reg0000[1] == 0)
		{
			reg0000[2] = 1;
			for(int i=0; i<3; i++)
			{
				delay(100);
				setDisp(no);
			}
		}

		if(indexbuf > 7)
		{
			tx_mode;
			if(CRC(buffer, indexbuf-2, 0) == 1 && buffer[0] ==
dev_address)
				checkframe(buffer);
			indexbuf = 0;
```

```
                    rx_mode;
                }
            }
}

void clk()
{
        PORTB |= (1<<PB7);
        PORTB &= ~(1<<PB7);
}

void setDisp(unsigned int angka)
{
        int num[4];
        unsigned char data[10] =
                                        { 0b00111111, 0b00000110, 0b01011011,
0b01001111,
                                          0b01100110, 0b01101101, 0b01111101,
0b00000111,
                                          0b01111111, 0b01101111 };
        unsigned char spin = 0b00000001;

        num[0] = (angka % 10);
        num[1] = (angka / 10) % 10;
        num[2] = (angka / 100) % 10;
        num[3] = (angka / 1000);

        for(int j = 0; j < 4; j++)
        {
            PORTB |= spin;
            unsigned char sendbyte = data[num[j]];

            for(int i = 0; i < 8; i++)
            {
                    if(sendbyte & 0b10000000) PORTB &= ~(1<<PB4);
                    else PORTB |= (1<<PB4);

                    clk();
                    sendbyte <<= 1;
            }

            PORTB |= (1<<PB6);
            PORTB &= ~(1<<PB6);
```

```
            delay(1);
            PORTB &= ~spin;
            spin <<= 1;
        }
}

unsigned long CRC(char message[], int panjangdata, int mode)
{
   unsigned char i,j;
   unsigned int CRCFull = 0xFFFF;
   unsigned int CRCH = 0xFF;
   unsigned int CRCL = 0xFF;
   unsigned  char CRCLSB;

   for (i = 0; i < (panjangdata) ; i++) //nilai crc tidak dihitung
   {
      CRCFull = (CRCFull ^ message[i]);

      for ( j = 0; j < 8; j++)
      {
         CRCLSB = (CRCFull & 0x0001);   //ambil LSB
         CRCFull = ((CRCFull >> 1) & 0x7FFF);  // geser

         if (CRCLSB == 1)
                        CRCFull = CRCFull ^ 0xA001;
            }
   }

   CRCH = ((CRCFull >> 8) & 0xFF);
   CRCL = (CRCFull & 0xFF);

       if(mode == 0)
       {
            if(CRCH == message[7] && CRCL == message[6]) return 1;
            else return 0;
       }
       else
       {
            return CRCFull;
       }
}
```

```
void checkframe(char buff[])
{
	unsigned long crc;
	int error = 0;

	switch(buff[1]) //Function Code
	{
	    case 2: //Membaca Coil
			buff[5] = reg0000[(int)buff[3]];
			sprintf( temp, "%c%c%c%c%c%c",
					buff[0], buff[1], buff[2], buff[3],
					buff[4], buff[5]);

			reg0000[(int)buff[3]] = 0;
			break;

	    case 6: //akses write holding register
			if ((int)buff[3] < 8)
			{
				reg4000[(int)buff[3]] = buff[4]*256 + buff[5];

				no = reg4000[1];
			}
			break;
	}

	//send back the data
	sprintf( temp, "%c%c%c%c%c%c",
			 buff[0], buff[1], buff[2], buff[3],
			 buff[4], buff[5]);
	crc = CRC(temp, 6, 1);
	buff[7] = (char)(crc>>8);
	buff[6] = (char)(crc);

	for(int i=0; i<8; i++)
	{
	    uart_putc(buff[i]);
	    setDisp(no);
	}
}
```

A.2 *Listing* Program Pascal (Delphi)

```
unit SAntrian;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, CPortCtl, CPort, MPlayer, ExtCtrls, OleCtnrs, jpeg,
  Buttons, OoMisc, AdPort;

type
  TForm1 = class(TForm)
    ComPort: TComPort;
    GroupBox4: TGroupBox;
    Label7: TLabel;
    Memo1: TMemo;
    Button2: TButton;
    Button3: TButton;
    Edit1: TEdit;
    Button1: TButton;
    Button4: TButton;
    Timer1: TTimer;
    OleContainer1: TOleContainer;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Label3: TLabel;
    BitBtn1: TBitBtn;
    Timer2: TTimer;
    Memo2: TMemo;
    Label5: TLabel;
    MediaPlayer1: TMediaPlayer;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure delay(SleepZ : Integer);
```

```
    procedure Timer1Timer(Sender: TObject);
    procedure ComPortRxChar(Sender: TObject; Count: Integer);
    procedure Button3Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure MediaPlayer1Notify(Sender: TObject);
    procedure Timer2Timer(Sender: TObject);
    Procedure KirimData(add, fcode, reg, rval : integer);
    function play(videoname : String): String;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  dreceived, creceived, dtemp: String;
  ireceived: integer;
  next : array of integer;
  loket : array [1..4] of integer;
  urutan, current, temp , mode: integer;
  add, fc, reg, dat, crcl, crch: integer;
  waitresponse, retry, vidfile : integer;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  mode := 0;
  waitresponse := 0;
  vidfile := 1;
  ireceived := 0;

  play(inttostr(vidfile) + '.wmv');
  MediaPlayer1.Notify := True;//ensures we are notified when song
completes
```

```
end;

procedure TForm1.MediaPlayer1Notify(Sender: TObject);
begin
  if MediaPlayer1.NotifyValue=nvSuccessful then
  begin
    MediaPlayer1.Stop;
    MediaPlayer1.Close;
    //vidfile := vidfile + 1;
    //if vidfile > 2 then vidfile := 2;

    //restart the song
    play(inttostr(vidfile) + '.wmv');
  end;
end;

function TForm1.play(videoname : String): String;
var
  Size : TRect;
  ScaleFactor : real;
begin
  MediaPlayer1.DeviceType := dtAutoSelect;
  MediaPlayer1.FileName := inttostr(vidfile) + '.wmv';
  MediaPlayer1.Open;

  (*init mediaplayer, load a movie*)
  Size := MediaPlayer1.DisplayRect;

  (*calculate a scaled format*)
  with Size do
  begin
    ScaleFactor := OleContainer1.Width / Right;
    Right := round(Right * ScaleFactor);
    Bottom := round(Bottom * ScaleFactor);
  end;

  MediaPlayer1.Display := OleContainer1;
  MediaPlayer1.DisplayRect := size;
  MediaPlayer1.Play;
```

```
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  if GroupBox4.Visible = True then GroupBox4.Visible := False
  else GroupBox4.Visible := True;
end;

procedure TForm1.delay(SleepZ : Integer);
var
  StartValue : LongInt;
begin
  StartValue := GetTickCount;
  While ((GetTickCount - StartValue) <= (SleepZ * 10)) do
  Application.ProcessMessages;
end;

function CRC16(Buffer:String; len: integer):Cardinal;
var
  i,j, CRCLSB, CRCFull: Integer;
begin
  CRCFull := $FFFF;

  for i := 1 to len do
  begin
    CRCFull := CRCFull xor ord(buffer[i]);

    for j := 0 to 7 do
    begin
      CRCLSB := CRCFull and $0001;
      CRCFull := (CRCFull shr 1) and $7FFF;

      if  CRCLSB <> 0 then CRCFull := CRCFull xor $A001;
    end;
  end;

  Result := CRCFull;
end;
```

```
Procedure TForm1.KirimData(add, fcode, reg, rval : integer);
var
  dstr: String;
  temp, k : integer;
begin
  dstr := chr(add) + chr(fcode) + chr(reg shr 8) + chr(reg) + chr(dat shr 8)
        + chr(rval);
  temp := CRC16(dstr, 6);
  dstr:= dstr + chr(temp) + chr(temp shr 8);

  memo2.clear;
  memo1.Clear;
  if comport.Connected = True then comport.writeStr(dstr);
  for k := 0 to 8 do memo2.text := memo2.text + inttostr(ord(dstr[k])) + ' ';
  timer2.enabled := True;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  if comport.Port <> '' then
  begin
    if (button2.Caption = 'Connect') then
    begin
      comport.Open;
      button2.Caption := 'Disconnect';
      button1.Enabled := true;
      button1.Caption := 'Stop';
      Timer1.Enabled := True;
    end else
    begin
      comport.Close;
      button1.Caption := 'Start';
      button1.Enabled := false;
      button2.Caption := 'Connect';
    end;
  end;
end;

procedure TForm1.Button4Click(Sender: TObject);
```

```
begin
  comport.ShowSetupDialog;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if (button1.Caption = 'Start') then
  begin
    button1.Caption := 'Stop';
    Timer1.Enabled := True;
  end else
  begin
    button1.Caption := 'Start';
    Timer1.Enabled := False;
  end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
var
  buff : String;
  CRCF, CRCL, CRCH, address, dvalue, fcode : Integer;
begin
  if (waitresponse = 0) and (ireceived < 8) then
  begin
    if mode = 0 then
    begin
      add := add + 1;
      if (add > 4) then add := 1;
      kirimData(add, 2, 1, 0);
      retry := 0;
    end else
    begin
      kirimData(add, 6, 1, loket[add]);
    end;
    waitresponse := 1;

    ireceived := 0;
    dreceived := '';
    creceived := '';
```

```
  end;

  if ireceived > 7 then
  begin
    CRCF := CRC16(creceived, 6);
    CRCH := (CRCF shr 8) and $FF;
    CRCL := CRCF and $FF;

    address := ord(creceived[1]);
    if (address = add) and (chr(CRCL) = creceived[7]) and (chr(CRCH) =
creceived[8]) then
    begin
      dvalue := (ord(creceived[5]) * 256) + ord(creceived[6]);
      fcode := ord(creceived[2]);

      if (dvalue = 1) and (fcode = 2) then
      begin
        loket[address] := loket[address] + 1;

        buff := format('%.3d', [loket[address]]);
        case address of
          1: label1.caption := buff;
          2: label2.caption := buff;
          3: label3.caption := buff;
          4: label4.caption := buff;
        end;
        mode := 1;
      end else if (dvalue = loket[address]) and (fcode = 6) then mode := 0;
    end;

    ireceived := 0;
    dreceived := '';
    creceived := '';
  end;
end;

procedure TForm1.Timer2Timer(Sender: TObject);
begin
  if waitresponse = 1 then
```

```
  begin
    waitresponse := 0;

    retry := retry + 1;
    add := add - 1;

    if retry < 10 then
    begin
      retry := 0;
      add := add + 1;
    end;
  end;
end;

procedure TForm1.ComPortRxChar(Sender: TObject; Count: Integer);
var
  str: string;
  k : integer;
begin
  comport.readStr(Str, 1);
  ireceived := ireceived + 1;
  creceived := creceived + str;
  dreceived := dreceived + inttostr(ord(str[1]));
  edit1.Text := inttostr(ireceived);

  if ireceived > 7 then
  begin
    ComPort.ClearBuffer(True, False);
    retry := 0;
    waitresponse := 0;

    memo2.Clear;
    memo1.clear;
    for k := 0 to 8 do memo1.text := memo1.text +
inttostr(ord(creceived[k])) + ' ';
  end;
end;

procedure TForm1.Button3Click(Sender: TObject);
```

```
begin
  Memo1.Clear;
  Memo2.Clear;

  ireceived := 0;
  dreceived := '';
  creceived := '';
end;

end.
```

--Halaman ini sengaja dikosongkan--

--Halaman ini sengaja dikosongkan--

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan data hasil uji implementasi protokol modbus pada sistem antrian terintegrasi ini diperoleh beberapa kesimpulan antara lain sebagai berikut:

1. Sistem yang dibangun memiliki kemampuan untuk mengelola data peserta berdasarkan nomor urut dari mulai pengambilan nomor hingga selesai. Sehingga petugas akan dapat lebih mudah dalam melakukan evaluasi pelayanan berdasarkan statistik dari tiap meja pelayanan.
2. Mikrokontroler AVR mampu menjalankan komunikasi dengan protokol modbus dengan cukup baik.

5.2 Saran

Terkait dengan kendala dan kekurangan dalam penyusunan tugas akhir ini, ada beberapa hal yang dapat penulis sarankan untuk pengembangan selanjutnya. Antara lain sebagai berikut:

2. Penambahan fitur pada software back office seperti grafik, dan perhitungan statistik lainnya.
3. Perlu analisa dan riset lebih lanjut untuk jumlah perangkat yang melebihi 32 *devices* dan jarak yang lebih dari 1000m.

Demikian saran yang dapat penulis sampaikan. Semoga dapat bermanfaat untuk ke depannya.

--Halaman ini sengaja dikosongkan--

DAFTAR PUSTAKA

1. Modbus-IDA. Modbus Application Protocol Specification V1.1b. 2006.
2. Modbus-IDA. MODBUS over Serial Line Specification and Implementation Guide V1.02. 2006.
3. Wardhana, Lingga. Belajar Sendiri Mikrokontroler AVR Seri ATMega8535 Simulasi, Hardware dan Aplikasi. Jogjakarta, Penerbit Andi, 2006.
4. Bies, Lammert. "RS485 Serial Information" <URL:http://www.lammertbies.nl/comm/info/RS-485.html>, Mei,2016
5. Prihati, Yani. Simulasi Dan Permodelan Sistem Antrian Pelanggan di Loket Pembayaran Rekening XYZ Semarang. Fakultas Ilmu Komputer Universitas AKI. 2012.

--Halaman ini sengaja dikosongkan--

DAFTAR RIWAYAT HIDUP

Septian Dwi Chandra. Lahir di Bojonegoro, 22 Septermber 1993. Putra dari Subiyanto dan Rokhani. Menamatkan pendidikan Sekolah Dasar di SDN Sukaresmi 06 pada tahun 2005. Di tahun yang sama meneruskan pendidikan di SMPN 04 Cikarang Utara hingga 2008. Setelah itu masuk di SMKN 01 Cikarang Selatan, jurusan Teknik Informatika dan Jaringan hingga 2011. Kemudian diterima di jurusan D3 Teknik Elektro Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, lulus pada tahun 2014 dan mengikuti program lintas jalur ke S1 Teknik Elektro di institusi yang sama, mengambil program studi Elektronika

E-mail :septian.d.chandra@gmail.com